Network Performance Impact of a Lightweight Linux for Cray XT3 Compute Nodes

Trammell Hudson (hudson@osresearch.net)
Ron Brightwell (rbbrigh@sandia.gov

1 Introduction

The Catamount Lightweight Kernel (LWK) [2] is the native compute node operating system for the Cray XT3 Redstorm system at Sandia National Labs. It was custom developed for the massively parallel compute node environment and is very simple compared to a modern operating system. Each node runs a single user application that is allocated all of the memory on the node as a physically contiguous block of pages. The operating system has no device drivers other than the SeaStar high-speed mesh interconnect. Other than the user application there is a privileged process control thread (PCT) that runs at 1 Hz to handle bookkeeping and a 10 Hz timer interrupt. The Portals message passing system used for communication between compute nodes is described in [1].

The stock Linux kernel used was a SuSE 2.6.5 with a 1000 Hz timer interrupt, 60 ms time slice and running a complete Linux runtime environment. Since these features have been identified as problematic [3], a special Linux kernel was built that ran with a 1 second time slice, a 17 Hz timer interrupt, no interactive runtime and only two processes: the Linux port of the PCT and the user application.

With some manipulation of the object files, it was possible to convert the Catamount user executables into Linux binaries that could run in this custom Linux environment without any further changes. This allowed the benchmarks to focus on only the part of the system that differed: the operating system.

2 CPU Availability

The Sandia Selfish benchmark is a reimplementation of the Zepto-OS Selfish benchmark [4]. The user application spins in a tight loop sampling the CPU's timestamp counter until it sees a discontinuity greater than some threshold due to an interrupt, a process scheduling event or any other system activity that takes the CPU away from the application. The current time and the magnitude of the step are recorded and are plotted as a time series in Figure 2 to graphically show the number of times the CPU is taken away from productive user work. Every interruption may cause TLB entries to be reused, cache lines to be flushed and contributes to the "rogue-OS" effect.

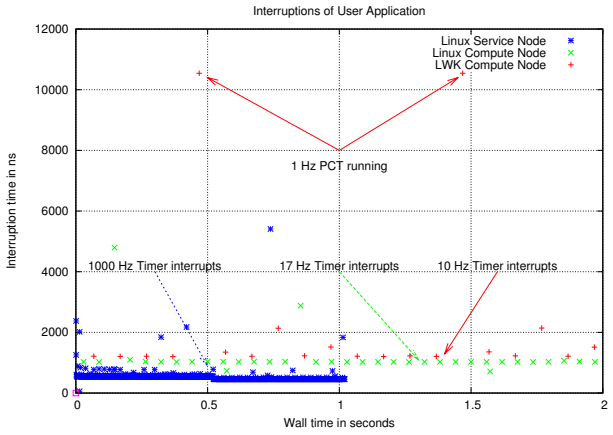

Figure 1: CPU Availability comparison

3 Message passing latency

One major difference in latency between Linux and Catamount is due to the event delivery mechanisms. Since Catamount is a single-programmed operating system, the user application can spin wait for an event to be delivered into user space. With a multiprogrammed Linux environment, however, this would adversely affect any other processes on the node, so the Portals device driver is able to put a process to sleep when it calls `select` on the file descriptor. When any event arrives the process is awakened and it checks the event queue to determine if its desired event has arrived.

Because compute node Linux is single programmed like Catamount, we have modified the device driver to allow the application to busy wait for incoming events. This is graphically shown in Figure 2 as the difference between the two Linux lines. This saves roughly 1.8 μs for the zero-byte latency.

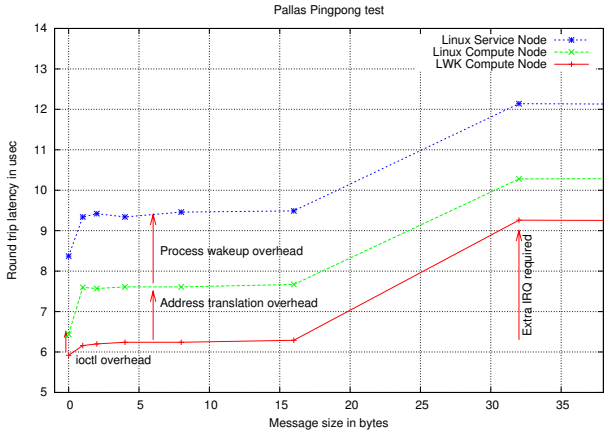

Figure 2: Latency comparison

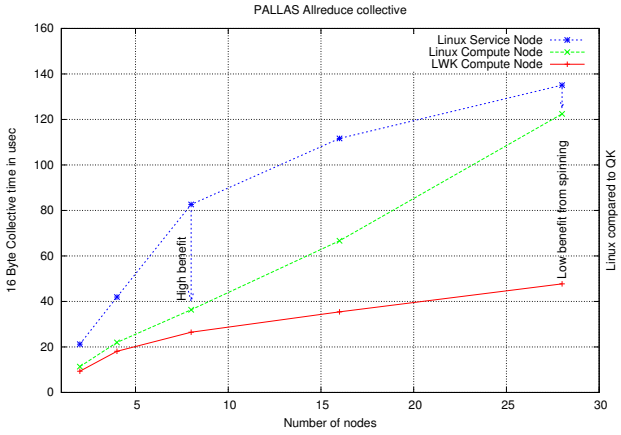

Figure 4: Allreduce comparison

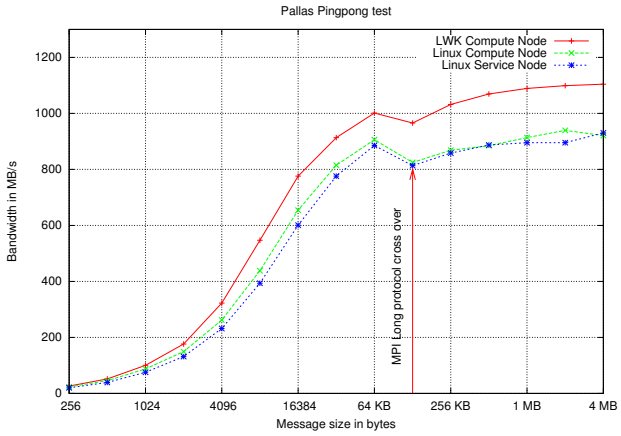

Figure 3: Bandwidth Comparison

4 Message passing bandwidth

The SeaStar DMA engines are capable of sending or receiving 256 KB in a single transfer, which allows Catamount to use far fewer DMA entries for long message transfers since all memory is physically contiguous. Both forms of Linux are only capable of programming single pages into the DMA engines, so all transfers are in 4 KB chunks. This reduces the overall bandwidth available to the application, as shown in Figure 3.

5 Collective Scaling

While the single node compute performance of the compute node Linux is very close to that of Catamount, it does not scale as well. Even at small scales of 28 nodes the lightweight kernel is far outperforming Linux on collective operations. The Pallas `Allreduce` collective benchmark shows that on a small number of nodes the compute node Linux is much better than the stock Linux kernel and is very close to Catamount's performance, but as the number of nodes increases the two Linux kernel performances converge. This result is shown as the node scaling graph for the 16-byte operation in Figure 4. It is interesting that this "rogue-OS" effect is evident at such a small scale, especially since the compute node Linux runtime has less interruptions than the Catamount kernel.

References

[1] Ron Brightwell, Kevin Pedretti, and Keith Underwood. Initial performance evaluation of the Cray SeaStar interconnect. In *Proceedings of the 13th IEEE Symposium on High-Performance Interconnects*, August 2005.

[2] Suzanne M. Kelly and Ron Brightwell. Software architecture of the light weight kernel, Catamount. In *Proceedings of the 2005 Cray User Group Annual Technical Conference*, May 2005.

[3] Fabrizio Petrini, Darren J. Kerbyson, and Scott Pakin. The case of the missing supercomputer performance: Identifying and eliminating the performance variability on the ASCI Q machine. In *Proceedings of the 2003 Conference on High Performance Networking and Computing*, November 2003.

[4] ZeptoOS. Zeptoos: The small linux for big computers. `http://www-unix.mcs.anl.gov/zeptoos/`.